NF01 – Printemps 2017

Examen Final

ATTENTION !
Utilisez quatre copies séparées, une par partie.

Seul le polycopié de cours est autorisé.

1ère Partie : Gestion d'un annuaire *(5 points)*

Vous devez proposer un programme en Pascal de gestion d'un annuaire. Pour chaque personne stockée dans l'annuaire, les informations suivantes sont renseignées : nom, prénom, numéro (dans la rue), nom de la rue, numéro de téléphone, code postal, ville. Votre programme doit permettre la saisie des informations, des recherches et des extractions suivant des critères choisis par l'utilisateur.

1. Définissez le type correspondant à une personne à l'aide des informations citées précédemment. Définissez l'annuaire en tant que tableau de taille maximum 500 contenant ce type de données.
2. Votre programme de gestion de l'annuaire doit contenir certaines fonctionnalités structurées sous forme de fonctions et procédures appelées dans le programme principal. Vous devez définir :
    a. Une fonction *taille_tab*, sans arguments, qui retourne la taille réelle du tableau demandée à l'utilisateur.
    b. Une procédure *saisie_tab* à deux arguments : l'annuaire et sa taille. Cette procédure permet la saisie de toutes les données pour toutes les entrées de l'annuaire. Ces données sont destinées à être exploitées dans le programme principal.
    c. Une fonction *critere_recherche*, sans arguments, qui permet à l'utilisateur de choisir le critère de recherche (nom, prénom, numéro, nom de la rue, numéro de téléphone, code postal, ville). Cette fonction doit retourner le choix de l'utilisateur.
    d. Une fonction *recherche* à trois arguments : le tableau annuaire, sa taille et le critère de recherche (Cf. 2.c). L'utilisateur doit pouvoir donner la valeur de recherche (par exemple Compiègne si le critère de recherche est 'ville'). Cette fonction doit retourner un tableau de la taille du tableau annuaire contenant la valeur 'v' pour les entrées de l'annuaire qui correspondent à la recherche demandée.
    e. Une procédure *affiche_tab* à trois arguments : le tableau annuaire, sa taille et un tableau du type de celui retourné par la fonction *recherche* précédente. Toutes les informations relatives aux personnes correspondant au critère de recherche doivent être affichées à l'écran.

```
program final2017;
const
   T_max = 50;
type
   ligne = string[40];
   typepersonne = record
                  nom, prenom, rue, ville:ligne;
                  num_tel, num_rue, code :ligne;
                 end;
   pers = array[1..T_max]of typepersonne;
   Nb_affiche = array [1.. T_max] of char;

var
  tab : pers;
  T_r, ind1 : integer;
  T_v : Nb_affiche;
  rech : string;

function taille_tab : integer;
var temp : integer;
begin
   repeat
       writeln('Taille du tablleau < ', T_max);
       read (temp);
   until (temp in [1..T_max]);
   taille_tab := temp;
end;

procedure ecrire_tab( var t : pers; s : integer);
begin
   for ind1 := 1 to s do
      with t[ind1] do
         begin
            writeln ('nom, prenom,N° rue, Nom rue, N° téléphone, Code, Ville');
            read (nom, prenom, num_rue, rue, num_tel, code, Ville);
            readln;
            readln;

         end;
end;

function critere_recherche : ligne;
var rep : char;
begin
   writeln ('N-nom,P-prenom,M-N°rue,R-Nom rue,T-N°téléphone,C-Code,V-Ville, Q-quitter');
   read (rep);
   if rep = 'Q' then
               critere_recherche := 'quitter'
       else
       case rep of
          'N'   : critere_recherche := 'nom';
          'P'   : critere_recherche := 'prenom';
```

```
            'M'   : critere_recherche := 'num_rue';
            'R'   : critere_recherche := 'rue';
            'T'   : critere_recherche := 'num_tel';
            'C'   : critere_recherche := 'code';
            'V'   : critere_recherche := 'ville';
          end;
 end;

function recherche (t : pers; critere : ligne; s : integer) : Nb_affiche;
var imprimer : boolean;
   champ_rech : string;
begin
     writeln('champ de recherche');
     read(champ_rech);
     readln;
     readln;
     for ind1 := 1 to s do
        begin
            with t[ind1] do
            begin
                if critere='nom' then imprimer := nom = champ_rech;
                if critere='prenom' then imprimer := prenom = champ_rech;
                if critere='num_rue' then imprimer := num_rue = champ_rech;
                if critere='rue' then imprimer := rue = champ_rech;
                if critere='num_tel' then imprimer := num_rue = champ_rech;
                if critere='code' then imprimer := code = champ_rech;
                if critere='ville' then imprimer := ville = champ_rech;
             end;

             if imprimer then recherche [ind1] := 'v';
            end;
end;


procedure affiche_tab (t:pers; v : Nb_affiche; s : integer);
begin
   for ind1 := 1 to s do
    begin
      if v[ind1] = 'v' then
                with t[ind1] do
                  begin
                    writeln (nom, prenom, num_rue, rue, num_tel, code, ville);
                  end;
    end;
end;

Begin
    T_r := taille_tab;
    for ind1:=1 to T_r do T_v [ind1]:=' ';
    ecrire_tab(tab,T_r);
    rech:=critere_recherche;
    T_v:=recherche (tab,rech,T_r);
```

```
   affiche_tab(tab,T_v,T_r);
end.
```

{ ---------- pensez à changer de copie ---------- }

2ème Partie : Récursivité *(5 points)*

2. 1 : La fonction itérative suivante, `divin(a,b)`, retourne le résultat de la division entière de *a* par *b* :

```
function divin(a,b : integer):integer;
begin
   divin := 0;
   while a >= b do
     begin
       divin := divin + 1;
       a := a - b;
     end;
end;
```

Il s'agit en fait du calcul `a div b` en Pascal.

Vous devez écrire la version récursive de cette fonction. Pour cela :

a) Déterminez l'expression récursive et le critère d'arrêt de la séquence des appels récursifs.
b) Ecrivez la fonction récursive `divinRec(a,b)` qui fournit le résultat de `a div b`, en utilisant uniquement les opérateurs "+" et/ou "-".

a) expression récursive : divin <- 1 + divin(a-b,b)

critère d'arrêt : si a < b alors divin <- 0

b)

```
function divinRec (a,b : integer):integer;
begin
  if a < b then
    divinRec := 0
  else
    divinRec := 1 + divinRec(a-b,b);
end;
```

2.2 : On souhaite écrire une fonction récursive prenant en paramètres deux valeurs, a un réel et b un entier, et retournant la valeur réelle a^b

a) Quel(s) critère(s) d'arrêt proposez-vous ?
b) Ecrivez la fonction

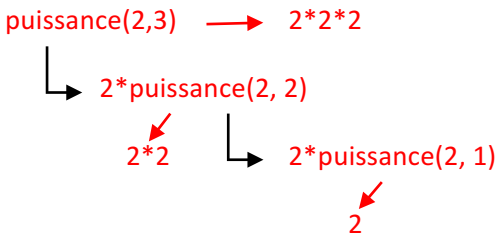
c) Combien y a-t-il d'appels récursifs pour *a=2* et *b=3* ? Décrivez graphiquement les appels récursifs dans ce cas.

a) On fera une condition sur b, l'exposant, s'il est égal à 1 la fonction retourne a, s'il est égal à 0 la fonction retourne 1, sinon elle fera l'appel récursif.

b)

```
function puissance(a: real, b:integer): real;
begin
        if b = 1 then {critère d'arrêt b=1 ou b=0}
                puissance := a
        else
                if b = 0 then
                        puissance := 1
                else
                        if b < 0 then
                                puissance := puissance(1/a,-b)
                        else
                                puissance := a*puissance(a,b-1);
end;
```

c) La sortie puissance(2, 3) est 8, avec 3 appels récursifs. Le dessin ci-dessous montrent les appels puis les retours successifs. Les flèches noirs sont les appels successifs (en plus de la toute première) et les flèches rouges les retours de ces appels



{ ---------- pensez à changer de copie ---------- }

3ème Partie : Fichiers de texte *(5 points)*

Dans cette partie, on suppose que l'on dispose d'un fichier texte composé exactement d'un mot par ligne, en minuscules et sans espacement. Chaque procédure ou fonction fait les opérations d'ouverture et de fermeture des fichiers dont le nom est passé en paramètre sous forme de chaine de caractères.

3.1 – Ecrire la fonction `NbMotsAvecVoyelle(nomf: string): integer;`

qui renvoie le nombre de mots commençant par une voyelle présents dans le fichier de nom `nomf`.

3.2 – On considère maintenant un fichier de nom `nomf1` trié par ordre lexicographique (*i.e.* alphabétique) croissant.

Ecrire la procédure `CompterChaqueMot(nomf1,nomf2: string);` qui écrit chaque mot du fichier `nomf1` de façon unique dans le fichier `nomf2`, suivi sur la même ligne d'un espace et du nombre d'occurrences de ce mot.

N.B. : La procédure ne fera qu'un seul passage sur le fichier `nomf1`. Celui-ci étant trié, on utilisera le fait que les occurrences d'un même mot sont forcément consécutives.

```
Function NombreMots (nomf : NomFichier) : integer;

Var  f : text;  s : ligne_t; n : integer;
Begin
 assign (f, nomf);
 reset (f);
 n := 0;
 while not eof (f) do
  begin
   readln (f, s);
   if s[1] in [a,e,I,o,u,y] > then
     n := n+1;
  end;
 close (f);
 NombreMots := n;
End;
```

```
Procedure CompterChaqueMot (nomf1, nomf2 : NomFichier);

Var f1, f2 : text;
     n : integer;
    lu, prec : ligne_t;
Begin
 assign (f1, nomf1);
 reset (f1);
 assign (f2, nomf2);
 rewrite (f2); n := 0;
 while not eof (f1) do
   begin
     readln (f1, lu);
     if n = 0 then
        n := 1 { init de lu puis prec }
     else if lu = prec  then
              n := n+1
          else begin { n est le nb d'occur. de prec, on l'écrit }
                  writeln (f2, prec, ' ', n);
                  n := 1;
               end;
     prec := lu;
```

```
  end;
{ On écrit le dernier mot, sauf s'il est vide }
if (n > 0) and (length (prec) > 0) then writeln (f2, prec, ' ', n);
close (f1);
close (f2);
end;
```

{ ---------- pensez à changer de copie ---------- }

4ème Partie : Retour vers le futur *(5 points)*

1) Ecrire en Pascal la fonction `RandomMinMax` qui renvoie un nombre pseudo-aléatoire compris entre `min` et `max` :

```
function RandomMinMax (min, max : integer): integer ;
```

On rappelle que la fonction Pascal `random(n)` retourne un nombre entier pseudo-aléatoire compris entre `0` et `n-1`.

2) On considère maintenant le programme `RetourVersleFutur` qui débute comme suit :

```
program RetourVersleFutur;
const
  N = 100;
type
  Person = record
             nom : string;      //nom de la personne
             annee : integer ;  //année actuelle de la personne
             temps : integer;   //temps (en secondes) nécessaire
                                //   pour revenir en 2017
           end;
  Tab = array[1..N] of Person;
var
  t : Tab;
```

Ecrire en Pascal une procédure `Saisie` permettant de saisir le nom des personnes du tableau `t`.
Le nombre de personnes n'est pas connu à l'avance mais sera inférieur strictement à 100. Ce tableau sera par la suite utilisé pour calculer les champs `annee` et `temps` de chaque personne.

3) Chaque personne doit choisir une période d'au moins 10 ans au cours de laquelle elle souhaite faire un voyage dans le temps. Le programme choisit aléatoirement une année de départ au cours de cette période.

Ecrire une procédure `CalculAnnee` en Pascal qui pour chaque personne du tableau `t` :

- demande dans quelle période, donnée par `min` et `max`, elle souhaite faire un voyage dans le temps. Cette période sera comprise entre -10 000 ans et 10 000 ans.
- appelle la fonction `RandomMinMax` et stocke l'année renvoyée par cette fonction dans le champ `annee` de la personne.

4) Le retour en l'an 2017 se fait également de manière aléatoire par essais successifs. Ecrire une procédure `CalculTemps` en Pascal qui, pour chaque personne du tableau `t`, calcule le temps nécessaire pour revenir en l'an 2017 et le stocke dans le champ `temps` de cette personne, sachant que :

- la période de saut est forcément de 10 ans autour de 2017, i.e. entre 2012 et 2022,
- le temps pour chaque saut nécessite 10 secondes.

5) Prendriez-vous ce risque ?

```
program RetourVersleFutur;
const
   N = 10;
type
  Person = record
     nom : string;
     annee, temps : integer;
     end;
  Tab = array[1..N] of Person;
var
 t : TAB;

procedure Saisie (var t : Tab);
var
 nb, i : integer;
begin
   repeat
       writeln('Combien d etudiants voulez vous saisir ');
       readln(nb);
   until (nb > 0) and (nb < N);
   for i := 1 to nb do
      begin
         write(' Etu ', i, ', Saisir nom : ');
         readln(t[i].nom);
         t[i].temps := 0;
      end;
   t[i+1].nom := '';
end;

procedure Affichage (t : Tab);
var
 i : integer;
begin
   writeln('---- Affichage tableau ---');
   i := 1;
   while (i < N) and (t[i].nom <> '') do
         begin
         with t[i] do
            write('Etu ', i, ' : ', nom, ', ', annee, ', ', temps);
         writeln;
         i := i + 1;
      end;
```

```
end;

function RandomMinMax (min, max : integer): integer ;
var
  range, annee : integer;
begin
    range := max - min;
    randomize();
    annee := min + random(range);
    //writeln('    Annee calculee : ', annee);
    RandomMinMax := annee;
end;


procedure CalculAnnee (t : Tab);
var
  min, max : integer;
var
  i : integer;
begin
    writeln('---- Calcul annee ---');

    i := 1;
    while (i < N) and (t[i].nom <> '') do
        begin
          with t[i] do
             begin
                writeln('Etu ', i, ' ', nom, ' : ');
                repeat
                     writeln('Choisir la periode min et max (au moins 10 ans entre min et max, min < max,
min > -300 000 et max < 10 000) : ');
                     readln(min); readln(max);
                until ((max-min) >= 10) and (min >= -300000) and (max <= 10000);
                annee := RandomMinMax (min, max);
             end;
             i := i + 1;
          end;
    Affichage(t);

end;

procedure CalculTempsRetour (t : Tab);
var
  min, max : integer;
var
  i : integer;
begin
    writeln('---- Calcul temps retour ---');
```

```
    writeln('Vous allez faire des sauts de temps sur un range de 50 ans autour de la date du jour pour
tenter de revenir...');
    writeln('Bonne chance');

    i := 1;
    min := 2012; max := 2022;
    while (i < N) and (t[i].nom <> '') do
        begin
          with t[i] do
             begin
                writeln('Etu ', i, ' ', nom, ' : ');
                while annee <> 2017 do
                    begin
                        annee := RandomMinMax (min, max);
                        temps := temps + 10;
                        //write('temps : ', temps, ', ');
                        //readln;
                    end;
                writeln('    Temps : ', temps, ', ');
             end;
             readln;
             i := i + 1;
         end;
    Affichage(t);

end;

begin
   Saisie(t);
   Affichage(t);
   CalculAnnee(t);
   CalculTempsRetour(t);
   readln;
end.
```